

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently amended) A computer implemented method for removing a part of a computing process comprising items of data, program code and executing states, the method comprising:

~~wherein said process splits splitting said computing process~~ into a first process and a second sub-process, the second sub-process comprising a portion of the program code and/or a portion of the execution states of said computing process not required by said first process[.]; and

~~wherein a construct is formed for storing forming a construct which stores~~ said second sub-process, while said first process retains the process identity of said computing process, and

wherein said second sub-process is a suspended dormant process comprising data, program code and execution states of said computing process temporarily not required by said first process, and

wherein said first process is able to re-acquire data, program codes and execution states from said suspended dormant process as and when required by said first process.

2. (Currently amended) A computer implemented method as claimed in claim 1 wherein said sub-process further comprises items of data of said computing process.

3. (Canceled)

4. (Currently amended) A computer implemented method as claimed in claim 1 wherein said construct is formed by a construct operation that suspends all active threads of said computing process and creates a new sub-process comprising at least some of the data and/or program modules and/or execution state of said computing process, and stores said sub-process in a data area of said computing process.

5. (Currently amended) A computer implemented method as claimed in claim 4 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

6. (Currently amended) A computer implemented method as claimed in claim 1 wherein said construct is provided with an authorizing signature.

7. (Currently amended) A computer implemented method as claimed in claim 1 wherein said construct is sent to a memory storage device.

8. (Canceled)

9. (Currently amended) A computer implemented method as claimed in claim 1 wherein said first process re-acquires data, program code and execution states from said suspended dormant process falling within specified ranges.

10. (Currently amended) A computer implemented method as claimed in claim 1 wherein said second sub-process comprises data, program code and execution states specific to a given user of a computing means, and wherein said second sub-process is formed to temporarily store said data, program code and execution states while said user is absent from said computing means.

11. (Currently amended) A computer implemented method as claimed in claim 1 wherein after said first process finishes executing data, program code and execution states from said first process are added to said second sub-process and said second sub-process is reactivated.

12. (Currently amended) A computer implemented method as claimed in claim 11 wherein said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said second sub-process prior to adding the program code and execution states from said first process to said second sub-process.

13. (Currently amended) A computer implemented method as claimed in claim 1 wherein a construct is formed for storing said second sub-process, while said first process is run in place of said computing process.

14. (Currently amended) A computer implemented method as claimed in claim 13 wherein said construct is formed by a construct operation that suspends all active threads of said computing process and creates a new sub-process comprising at least some of the data and/or program modules and/or execution state of said computing process, and stores said sub-process in a data area of said first process.

15. (Currently amended) A computer implemented method as claimed in claim 14 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

16. (Currently amended) A computer implemented method as claimed in claim 13 wherein said construct is provided with an authorizing signature.

17. (Currently amended) A computer implemented method as claimed in claim 13 wherein said construct is sent to a memory storage device.

18. (Canceled)

19. (Currently amended) A computer implemented method as claimed in claim 1 wherein said first process re-acquires data, program code and execution states from said suspended dormant process falling within specified ranges.

20. (Currently amended) A computer implemented method as claimed in claim 13 18 wherein said second sub-process comprises data, program code and execution states specific to a given user of a computing means, and wherein said second sub-process is formed to temporarily store said data, program code and execution states while said user is absent from said computing means.

21. (Currently amended) A computer implemented method as claimed in claim 13 wherein after said first process finishes executing data, program code and execution states from said first process are added to said second sub-process and said second sub-process is reactivated.

22. (Currently amended) A computer implemented method as claimed in claim 21 wherein said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said second sub-process prior to adding the program code and execution states from said first process to said second sub-process.

23. (Currently amended) A computer implemented method for removing a part of a computing process comprising items of data, program code and executing states, the method comprising:

~~wherein said process splits splitting said computing process into a first process and a second sub-process, the second sub-process comprising a portion of the program code and/or a portion of the execution states of said computing process not required by said first process[[],];~~

~~wherein a construct is formed for storing forming a construct which stores said second sub-process, while said first process retains the process identity of said computing process[[],];~~

~~wherein after said first process finishes executing data, adding program code and execution states from said first process are added to said second sub-process and reactivating said second sub-process is reactivated; and~~

~~wherein discarding by said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said second sub-process prior to adding the program code and execution states from said first process to said second sub-process.~~

24. (Currently amended) A computer implemented method for removing a part of a computing process comprising items of data, program code and executing states, the method comprising:

~~wherein said process splits splitting said computing process~~ into a first process and a second sub-process, the second sub-process comprising a portion of the program code and/or a portion of the execution states of said computing process not required by said first process[[],]; and

~~wherein a construct is formed for storing~~ forming a construct which stores said second sub-process, while said first process is run in place of said computing process, and

wherein said second sub-process is a suspended dormant process comprising data, program code and execution states of said computing process temporarily not required by said first process, and

wherein said first process is able to re-acquire data, program codes and execution states from suspended dormant process as and when required by said first process.

25. (Currently amended) A computer implemented method for removing a part of a computing process comprising items of data, program code and executing states, the method comprising:

~~wherein said process splits splitting said computing process~~ into a first process and a second sub-process, the second sub-process comprising a portion of the program code and/or a portion of the execution states of said computing process not required by said first process[[],];

P21105.A13

wherein a construct is formed for storing forming a construct which stores said second sub-process, while said first process is run in place of said computing process[[.]];

wherein after said first process finishes executing data, adding program code and execution states from said first process are added to said second sub-process and reactivating said second sub-process is reactivated; and

wherein discarding by said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said second sub-process prior to adding the program code and execution states from said first process to said second sub-process.